A Web-based Application for Writing Novels

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 60 (2015) 1014 – 1020

Procedia Computer Science

19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

A Web-based Application for Writing Novels

Shouhei Nishihara[a], Motoki Miura[b]

[a]*Department of Applied Science for Integrated System Engineering, Kyushu Institute of Technology, 1-1 Sensui, Tobata, Kitakyushu Fukuoka, 804-8550, Japan*
[b]*Faculty of Basic Sciences, Kyushu Institute of Technology, 1-1 Sensui, Tobata, Kitakyushu Fukuoka, 804-8550, Japan*

Abstract

In this paper, we propose a method for assisting amateur writers in novel writing. Amateur writers can publish their work intensively through web infrastructures. This situation is beneficial, because it encourages amateur writers to enhance their skills by sharing their work. However, writing a good novel is difficult for a novice, because the novel-writing task requires the management of many character settings and maintenance of consistency throughout the novel. The length of a novel increases the difficulty of the task and decreases motivation owing to cumbersome management tasks. To reduce the burden, we introduce automatic keyword suggestion and highlighting techniques. Automatic keyword suggestion finds names of characters from the draft text and proposes their addition to the character list. The character names in the list are automatically linked to the corresponding words in the draft text. Using such functions, amateur writers can easily check the consistency of their novels while writing, similar to the use of an integrated development environment for software development. We have implemented a web application that provides the functions, and we consider the effectiveness of the proposed method here.

Peer-review under responsibility of KES International

Keywords: Keyword Suggestion; Novice Writer Support; Web-based Editor;

1. Introduction

The recent rapid evolution of the Internet enables the publicizing of a person's creative work at low cost. As a result, various creative activities, such as writing novels, have become popular among amateurs. Amateur novelists post their work on their personal sites, bulletin boards, or comprehensive sites, increasing the total number of novels on the web. However, writing a novel is difficult. Novelists must simultaneously consider many factors, such as determining details of the story, selecting suitable words, and managing settings. Moreover, novelists review their written text frequently, because it might include mistakes or inconsistencies. As the text of the novel lengthens, the reviews become more difficult, complicated, and time consuming. The time and effort of novelists is better used for

* Corresponding author. Tel.: +81-80-6446-8622.
E-mail address: nisisyou24@gmail.com

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2015.08.144

creative writing tasks, if possible, and not for consistency checks and character management. This applies especially to amateur writers, who have limited time to spare for writing.

To maximize efficiency, we propose a method that assists amateur writers in writing novels. The method contains functions of automatic keyword suggestion and highlighting derived from an integrated development environment (IDE). By applying these functions to writing novels, amateur writers can reduce the burden of consistency checks and character management. In addition, we have developed a web application that provides these functions. Using the framework of the web application, amateur writers can utilize the functions immediately, without installing new software.

The structure of the paper is as follows. In section 2, we describe the method we propose in detail. Section 3 describes the implementation of the web application that realizes the proposed method. In section 4, we review related work regarding the use of software to write or generate novels. We conclude and discuss future work in section 5.

2. Proposed method

In this section, we describe our proposed method to reduce the burdens of consistency checks and character management for amateur novelists. The method contains functions for (1) automatic keyword extraction, (2) keyword highlighting, and (3) commenting. Using these functions, amateur writers can reduce the time and effort of those tasks.

2.1. Automatic keyword extraction

First, we describe automatic keyword extraction. The purpose of automatic keyword extraction is to facilitate writers management of characters and to highlight keywords while editing the main text. This assists writers in maintaining consistency of character settings and preferences.

The function extracts candidates for character names as follows. Since the names of characters tend to appear frequently in the main text of the novel and consist primarily of proper nouns, the function identifies proper nouns appearing in the main text.

Novels typically include keywords, which are used frequently, primarily with a novel's setting. If a name has been used in the text, the character setting is used in the scenario. Therefore, automatically identifying keywords can reduce the labor required for the confirmation of a novel's setting. We expect that frequent confirmation will reduce conflicts. To extract proper nouns, we use Mecab[1] a morphological analyzer. Mecab recognizes Japanese text and outputs the types of the words. Through the use of Mecab, proper nouns can be extracted as candidates for character names.

2.2. Character management

Candidates for character names extracted using automatic keyword extraction are shown on the display. When an amateur writer selects a candidate, the word is registered to a character table. The character table provides functions for managing each character, which contain the name and preferences, such as sex, year, and personality profile. These preferences are to be inputted by amateur writers themselves at the time of registration, or afterward. Since the preferences are always shown in the display, amateur writers can check them at any time while writing the main text.

2.3. Keyword highlighting

To ease the writers' checking of registered character names, we introduce a keyword highlighting function. The keyword highlighting function displays the registered character names in different styles of text in the main text editor. The text style contains the text and background colors, as specified by the writers.

2.4. Comment function

To maintain miscellaneous settings and preferences of supporting role characters, we introduced a comment function. The comment function enables writers to record miscellaneous notes within the main text. Since the comment

function is an important feature of an IDE, we included the function in the novel editor. Using the function can help amateur writers remember their ideas and thoughts when writing the text.

3. Implementation

We have developed a web application to implement the proposed method and the functions described above. In this section, we explain details of those functions through examples. Fig. 1 shows a screenshot of our prototype system. The system consists primarily of three parts, (1) scenario outline editor, (2) setting management table, and (3) main text editor.

Fig. 1. Prototype system.

First, the writer composes an outline of the scenarios using the scenario outline editor. Each row of the scenario outline editor represents a scenario unit (e.g., chapter). Some rows might have sub-rows (e.g., sections). As a result, the rows organize the outline of the novel. By selecting scenario units, writers can change the active scenario unit. Since the main text of the novel is managed separately in the scenario unit, the main text editor displays only content related to the active scenario unit. Settings that are highly related to the active scenario unit are displayed in a setting management table.

3.1. Automatic keyword extraction

Writers input the text in the main text editor as usual. The inputted text is stored in the scenario units. Keyword extraction begins, when the [auto table] button is pressed. The prototype system transfers the text to a server-side script, which then processes the text using Mecab[1].

Mecab separates the text into words by considering the types of words. The script picks up only nouns and noun phrases, counts the occurrences, and returns the words and phrases list sorted by frequency.

Fig. 2 shows the keywords extracted from “Run, Melos!,” a famous story in Japan. The count represents the number of occurrences in the text. In this case, the count of the hero “Melos” was 75. However, some unnecessary words also appeared in the list. To remove the unnecessary words, we introduced some stop-words that are not to be included in the list. Figure 3 shows the results of the modification. We consider that the stop-words can be managed semi-automatically by introducing extra rules. For example, some unnecessary words can be detected as being single Hiragana letters (in Fig. 2 (1)), general nouns (Fig. 2 (2)), or non-independent words (Fig. 2 (3)).

count	name
78	の
76	私
75	メロス
30	人
20	おまえ
19	王
18	友
15	セリヌンティウス
15	事
13	君
13	よう

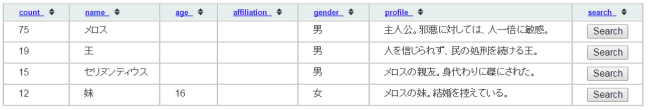
Fig. 2. Extracted keywords (before modification)

count	name
76	私
75	メロス
30	人
20	おまえ
19	王
18	友
15	セリヌンティウス
13	君
12	男
12	妹
11	市

Fig. 3. Extracted keywords (after modification)

Fig. 4 shows the interface for registering keywords from the extracted words. The number next to a word indicates the word's frequency. Writers can select words by clicking the corresponding check boxes. Only the words selected are imported into the setting management table. In Fig. 4, the writer selects four words to be imported.

Fig. 4. Interface for registering keywords



count	name	age	affiliation	gender	profile	search
75	メロス			男	主人公。邪悪に対しては、人一倍に敏感。	Search
19	王			男	人を信じられず、民の処刑を繰する王。	Search
15	セリヌンティウス			男	メロスの親友。身代わりに磔にされた。	Search
12	妹	16		女	メロスの妹。結婚を控えている。	Search

Fig. 5. Setting management table

Fig. 5 shows a snapshot of the setting management table after importing the selected words. The writer can manage the table by inputting preferences for each character. Moreover, the writer can define aliases (i.e., alternative names) for characters. For example, friend always represents Selinuntius, Melos's friend. In such a case, the word friend is an alias of Selinuntius. The setting management table provides a simple sorting function. A writer can reorder the list of characters by clicking column headers. When writers sort the list by frequency, they can review the key characters in that scenario.

3.2. Keyword highlighting

We have implemented a keyword highlighting function to assist writers in recognizing the appearance of characters in the main text. When writers click on a "Search" button displayed on the setting management table (see Fig. 6), the word is highlighted with different text and background colors (see Fig. 7). In the prototype system, the default colors of text and background are assigned automatically. When writers change the color setting, the behavior of the main text editor changes dynamically. The highlighting of the word is removed, when the button is clicked again. The dynamic highlighting of the main text editor was implemented using CodeMirror[2] a Javascript-based text editor. Fig. 8 illustrates dynamic behaviors of the main text editor. Once the highlighted word in the main text editor is

modified, the effect of the highlighting disappears. Conversely, when a writer inputs the word again, it is highlighted automatically by the main text editor.

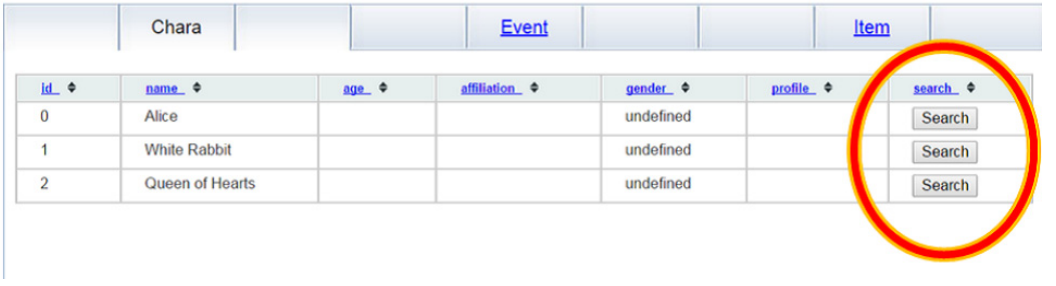


Fig. 6. Search button.

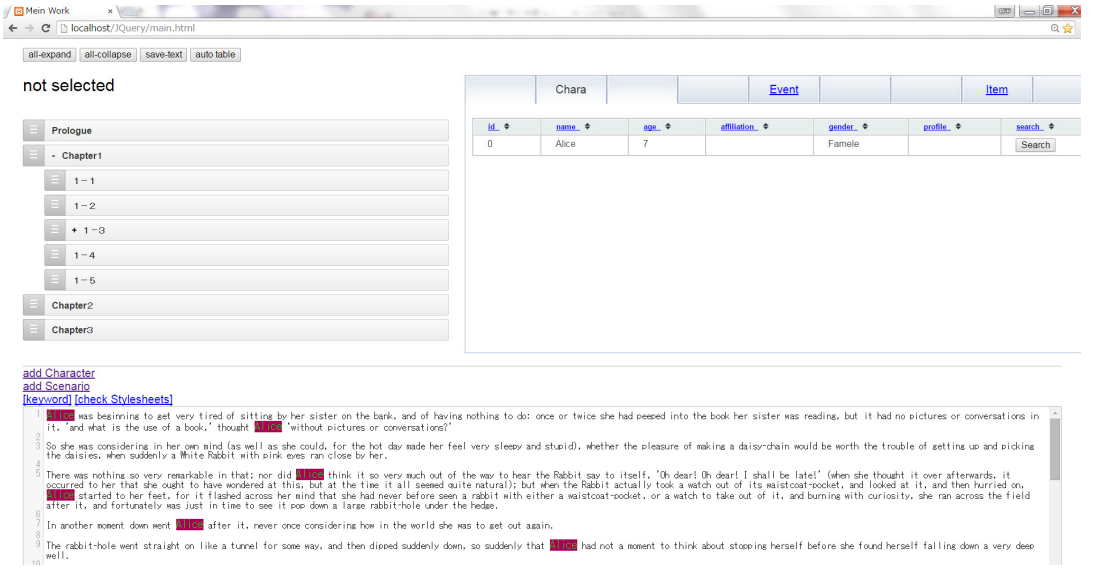


Fig. 7. Highlighted text.

3.3. Comment function

Fig. 9 shows a screenshot of the main text editor when it contains several comments. Writers can embed comments by surrounding the text with an at sign (@). When two at signs appear in the text, the remaining line becomes a comment. The regions of the comment are also highlighted with different colors in the main text editor.

The comment function is useful for remembering miscellaneous settings. If the story was sufficiently short, the writer can finish writing without interruption. However, it is difficult to maintain miscellaneous settings when writing a long story. In such a case, settings are typically written in other notebooks. However, management of the other notebooks increases the burden and time for referring to notes.

We consider that the comment function solves the issues of referring to notes by embedding them. The comment function can be used to improve the text by trial and error. Writers can easily switch two or more sentences by adding or removing the at signs.

For a writer who wants to check all the comments, we have developed a function to bundle all comments through the main text. At present, the system does not support the maintenance of consistency between the main text and the

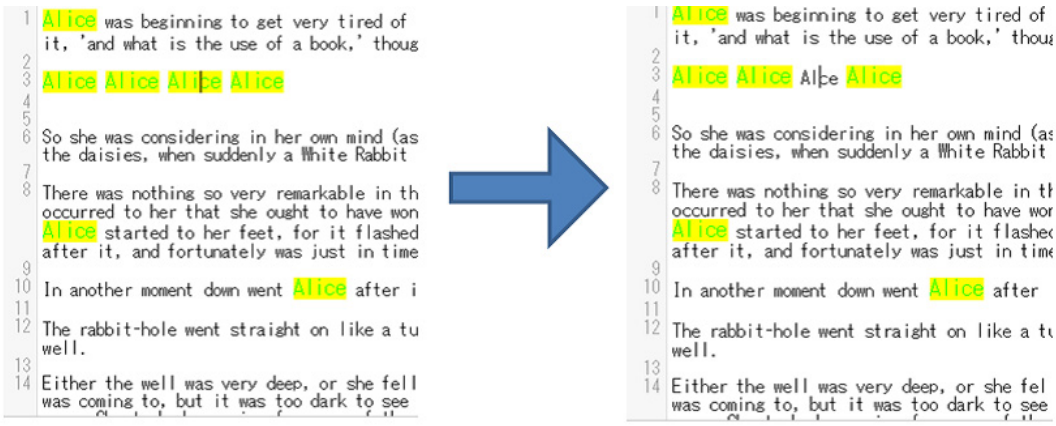


Fig. 8. Dynamic behavior of the main text editor.

contents of the comments. Consequently, writers are responsible for finding contradictions and maintaining consistency.

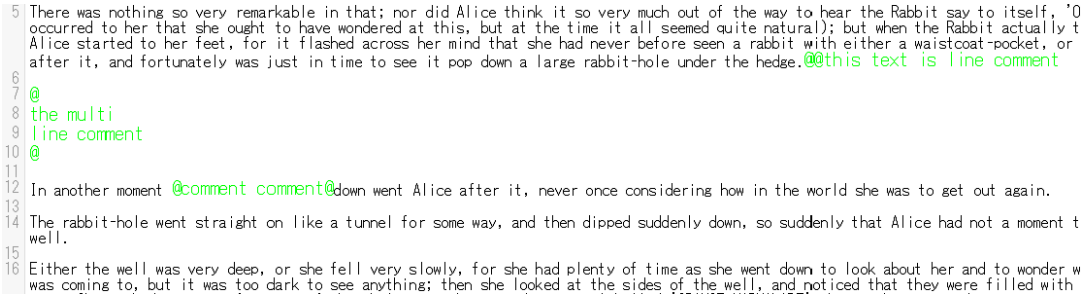


Fig. 9. Comment function.

4. Related works

Automation of novel creation has been extensively studied. BRUTUS[3] is a program that writes short stories on pre-defined themes such as betrayal. MINSTREL[4] is a case-based problem-solver that stores past cases in an episodic memory. MEXICA[5] is a computer model based on an engagement-reflection cognitive account of creative writing that produces stories. These systems automatically produce text using their respective methods. Such automatic writing might be of help when amateur writers begin writing prologs, one of the most difficult tasks. However, the automatic writing approach can reduce the creativity of amateur writers. Therefore, we recommend that the use of automatic text generation be limited.

Literate programming[6] is an approach to programming introduced by Donald Knuth in which a program is given as an explanation of the program logic in a natural language such as English. We adopted the literate programming approach for writing novels and incorporated it into our web-based system.

Spell checking[7] is a fundamental method for reducing misspelling. The result of spell checking is shown by text with highlighting, such as underlining. For programming editors, syntax highlighting is also popular for reducing errors and recognizing structure. Therefore, we expect that these features can be used to support novel writing.

5. Conclusion and Future Work

In this paper, we proposed a method and implemented a web application to assist amateur writers in writing novels. In particular, we introduced automatic keyword suggestion and highlighting techniques. The automatic keyword suggestion finds names of characters from the draft text and proposes them for adding to the character list. The selected keywords are highlighted in the main text editor. We also provided setting manager and comment functions to enrich the novel-writing environment. Using such functions, similar to the use of an IDE, amateur writers can easily check the consistency of their novels while writing. We hope that the web application contributes to the popularization of novel writing by amateurs.

This system was developed for writing a single novel. However, writers sometimes create a series of novels. As future work, we will consider how to share and apply the novels settings throughout a series of novels. We will also improve the accuracy of automatic keyword extraction using natural language processing. Once the accuracy of keyword extraction is improved, we will assess the performance of the system. To assess the performance, we will ask some writers to utilize the system and provide feedback. In addition, we will evaluate the usability of the system by reviewing tasks of existing novels.

References

1. Kudo, T.. Mecab: Yet another part-of-speech and morphological analyzer. http://mecab.sourceforge.net/; 2005.
2. Haverbeke, M.. Codemirror (version 2. x). 2011.
3. Bringsjord, S., Ferrucci, D.. *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine*. Psychology Press; 1999.
4. Turner, S.R.. Minstrel: a computer model of creativity and storytelling 1993;.
5. PÉrez, R.P.Ý., Sharples, M.. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence* 2001;**13**(2):119–139.
6. Knuth, D.E.. Literate programming. *The Computer Journal* 1984;**27**(2):97–111.
7. Zhao, Y., Truemper, K.. Effective spell checking by learning user behavior. *Applied Artificial Intelligence* 1999;**13**(8):725–742.

Acknowledgements

The part of this research was supported by the fund of Telecommunication Advancement Foundation and JSPS KAKENHI Grant-in-Aid for Scientific Research (C): Grant Number 15K00485.